OPEN ACCESS

ISAR Journal of Multidisciplinary Research and Studies

Volume 1, Issue 1, 2023 | Page: 72-79

Abbriviate Title- ISAR J Mul Res Stud

ISSN *(Online)*- 2583-9705

https://isarpublisher.com/journal/isarjmrs

Blockchain in Software Engineering: Secure and Decentralized Solutions

Santosh Panendra Bandaru*

Independent Researcher, USA.

***Corresponding Author**
Santosh Panendra Bandaru

Independent Researcher, USA.

Article History
Received: 23.04.2023
Accepted: 01.06.2023
Published: 31.07.2023

Abstract: In recent years, the use of blockchain-based technology has increased dramatically, transforming a number of businesses. Knowing the software development methodologies used by blockchain developers is essential as blockchain-based solutions continues to gain traction. The experiences of blockchain developers using Software Engineering (SE) methodologies in blockchain development are investigated in this research study. The relevance of various phases in the Software Development Life Cycle (SDLC) for projects related to blockchain, the value of SE techniques, and preferred methodology were among the subjects of an online poll that we launched. In recent years, the Blockchain and its implementations, known as Smart Contracts, have been more popular across all industries that need reliability and strong certifications. Some have even gone so far as to claim that the "Blockchain revolution" is comparable to the early days of the Internet & the Web. Because of this, the amount of software development centred on Blockchain technology is expanding at an astounding pace. Many software engineers feel that the high level of interest in Blockchain technologies is leading to unregulated and rushed software development, a kind of first-come, first-served competition that does not guarantee software quality or that the fundamentals of engineering software are taken into consideration. The improvement of security and trust is one of blockchain's most important contributions to software development. The decentralised nature of blockchain technology naturally reduces the danger of single points of failures that are common in traditional centralised systems. The immutable ledger promotes trust between users and developers by guaranteeing that data cannot be changed without consensus after it has been recorded. The procedure is founded on a number of Agile concepts, including incremental and iterative development and User Stories. But it also uses more formal notations, including certain UML diagrams that describe the system's architecture, with enhancements to express particular Blockchain development principles. A thorough explanation of the procedure is provided, along with an example to demonstrate its operation.

Keywords: UML Diagrams, Blockchain Projects, Agile Practices, Software Engineers, Blockchain Developers, Thereby Fostering, Blockchain Revolution, Security and Trust.

Cite this article:

Bandaru, S. P., (2023). Blockchain in Software Engineering: Secure and Decentralized Solutions. *ISAR Journal of Multidisciplinary Research and Studies*, *1*(1), 72-79.

I. Introduction

Blockchain technology has become a disruptive force that has the ability to completely change a number of sectors, including governance, healthcare, supply chain management, and financing. It's decentralised, transparent, immutable, [1, 2] nature opens up new avenues for safe and effective transaction processing and data management. Understanding the software development methodologies used by developers is becoming more and more important as the use of blockchain-based solutions keeps expanding [2, 3].

Software engineering (SE) concepts and the technology that underlies blockchain must be well understood in order to design blockchain applications [3, 5]. Designing, developing, and deploying decentralised apps (D-Apps) while maintaining security, privacy, and attack resistance provide special problems for blockchain developers. The minimising of attacks and vulnerability in smart contracts is one of the issues posed by the distributed characteristics of blockchain networks [2, 6]. Thus, assuring the dependability, security, and effectiveness of blockchain-based systems depends critically on comprehending the SE techniques used by developers of blockchain technologies and integrating SE concepts and processes into blockchain development.

One of the newest trends in software development is "decentralised applications," or "DApps." Typically, DApps operate on a blockchain, which was first developed to administer the digital money known as Bitcoin [5]. Due to its peer-to-peer node

architecture, blockchain software is inherently transparent, redundant, and decentralised.

Developers and management discovered that a blockchain may also be the perfect setting for a decentralised computer a few years afterwards the launch of Bitcoin in 2009 [5, 6]. Following Nick Szabo's proposal, this resulted in the creation of the Ethereum blockchain, which is a network that has nodes that can also execute Turing-complete programs known as "smart contracts" (SCs). Although they have certain unique properties, SCs are generic computer programs [7, 9]. Their initial concept was that they might be used to automatically enforce contractual commitments without relying on a central authority and without being limited by time or place.

A significant amount of capital poured into blockchain projects as a result of this interest, including some unrelated to virtual currency [9, 10]. These are the distributed ledgers (DL), often known as "permissioned" blockchains, which are meant to be operated by a group of nodes selected by invitation.

Since a DL may use different cryptographic techniques to retain the transaction history immutably (it is not dependent on a mechanism for consensus to pack the transaction history into a chain of blocks), bear in mind that while a blockchain is a kind of Digital Ledger Technology (DLT), and not all DLTs are built on top of a blockchain. We shall mostly discuss "blockchain" [11,12] in the following, however "DLT" is sometimes used interchangeably.

The development of software within a new paradigm is the foundation of all blockchain-related efforts, including new digital currencies that operate with their own blockchain technology, exchanges and other web-based businesses employing digital currency, Initial Coin Offering (ICO) start-ups, and apps operating on permissioned DLs. As is often the case with new waves of technology, we helped in this case [11,12] in an attempt to be the very first on the market. This resulted in rapid application development, sometimes at the expense of sound development procedures, thorough testing, and security evaluation.

Following a few catastrophes, digital currencies worth almost a billion dollars of USD (or close to it at the nominal exchange rate) were either lost or stolen [1, 6]. A number of exchanges were compromised, and smart contract features were often abused to capitalise on their novelty and the rushed software development process.

It is well recognised that using good Software Engineering (SE) principles and adhering to an explicit process of development are necessary to create a software system that is dependable and maintainable. Among them, we emphasise the significance of requirements elicitation, system design, particular notations, testing, and security assessment in the context of blockchain development [11, 16]. To put it simply, we need BOSE, or blockchain-oriented software engineering [16, 17].

The improvement of trust and security is one of blockchain's most important contributions to software development. Because blockchain technology is decentralised, it naturally reduces the danger of single points of failures that are common in traditional centralised systems. The immutable ledger promotes trust between users and developers by guaranteeing that data cannot be changed without consensus after it has been recorded [17, 18]. Software development has expanded thanks to the idea of smart contracts, which are self-executing agreements with the conditions of the agreement encoded directly into the code [19, 20].

These programmable contracts guarantee that agreements are carried out as intended, automate procedures, and eliminate the need for middlemen. In fields like the management of supply chains, where blockchain may provide software project transparency and traceability, this automation is very advantageous. Additionally, blockchain brings about a paradigm change in teamwork and project management [20]. With a single, unchangeable version of data related to the project accessible to all stakeholders, blockchain's transparency may result in more effective project management. This openness facilitates version control and collaborative coding, guaranteeing that modifications are monitored and verified [22]. Additionally, blockchain may be very helpful in managing software licensing and safeguarding intellectual property, offering a transparent and safe method of doing so.

Agile methodology and blockchain integration provide a strong synergy. The capacity of blockchain technology to provide safe, transparent, [21,22] and unchangeable records of project progress might be advantageous for agile methodologies, which prioritise adaptability and iterative development. In addition to offering a strong foundation for agile project tracking, this connection may improve deployment and continuous integration procedures. Blockchain use in the development of software is not without its difficulties, but [22]. Significant obstacles may arise from blockchain technology's intricacy, scalability problems, and energy use [23]. Crucial issues that must be resolved include negotiating the regulatory environment and dealing with moral dilemmas like data privacy. Blockchain technology has a bright future in software development, despite these obstacles [11]. New developments show that blockchain is increasingly integrating with other technologies, such as DevOps and Artificial Intelligence (AI). A current field of study is how blockchain might improve AI-driven applications and facilitate machine-to-machine transactions. Likewise, incorporating blockchain into DevOps procedures might result in development pipelines that are safer and more effective.

II. Related Work

- ✓ **Fundamentals of Blockchain:** The Architecture of Blockchain The fundamental framework that establishes how blockchain functions is known as blockchain architecture. A timestamp, [16], transaction data, and the cryptographic hash relating to the preceding block are all included in each of the blocks that make up this system. The data stored on the blockchain is guaranteed to be transparent and unchangeable because to its structure. Additionally, distributed databases are included into the design, which offers benefits over centrally hosted databases by improving security and lowering the possibility of data tampering.
- ✓ **Key Features of Blockchain:** Decentralisation, transparency, immutability, & security are among the salient characteristics of blockchain technology. By preventing any one party from controlling the whole network, decentralised administration lowers the possibility of manipulation and boosts participant trust

[14]. The public ledger, which makes all transactions transparent to participants, is how transparency is accomplished. Immutability guarantees that information cannot be changed without network consensus after it has been entered into the blockchain [16]. Consensus procedures such as Proof of Stake (PoS) and Proof of Work (PoW) and cryptographic algorithms improve security.

- ✓ **Types of Blockchain Networks:** The two main categories of blockchain networks are permissionless & permissioned [14]. Public blockchains, also referred to as permissionless blockchains, let everyone join without needing authorisation. Ethereum and Bitcoin are two examples. Permissioned blockchains, on the other hand, limit access and demand consent before users may join the network. These frequently have applications in organisational and business contexts. Furthermore, blockchain technology has been used in a number of domains, including as the Internet of Things (IoT), because it gives IoT devices protection and credibility [16].
- ✓ **Fundamentals of Software Development:** As technology has advanced and consumer demands have changed, software development has changed dramatically throughout time [16]. The basic elements of software development are examined in this part, along with its procedures, approaches, and the significance of interaction between humans and computers.
- ✓ **Software Development Process:** Large-scale and complicated software development relies heavily on the software development process, which is a methodical, disciplined, and measurable methodology [19]. To help stakeholders complete the final software products, a variety of software process approaches, including Agile & DevOps, have been employed. Iterative development, ongoing integration, and cooperation between the development and operations departments are highlighted in these approaches.
- ✓ **Human-Computer Interaction in Software Development:** A crucial component of software development, especially during the design and development stages, is Human-Computer Interaction (HCI) [18]. Developing software that is easy to use requires taking into account the behavioural, cognitive, perceptual, efficient, and physical aspects of human interaction. Software projects may become more successful, better, and easier to use by incorporating HCI concepts into the development process.
- ✓ **Software Development Life Cycle Methodologies:** The SDLC, or software development life cycle, includes a number of approaches that are intended for certain uses. Effective management of processes for development is essential for software development success as technological complexity rises [18]. Human-cantered software applications may be managed and delivered using a variety of methods provided by several SDLC models, including Waterfall, Agile, and Scrum.
- ✓ **Education and Industrial Needs in Software Development:** Research on the discrepancy between the training of software practitioners and industry demands is still underway. Research has shown gaps in software engineering education's covering of computer basics, human skills, software processes, & HCI [15]. In order to match educational curriculum with the changing demands of the computer software sector, these gaps must be filled.
- ✓ **Commit Frequency in Software Development:** Software development procedures need an understanding of commit frequency, which indicates how often a developer contributes code to a project [14]. Commit frequency analysis in open-source development of software sheds light on the distinctions between successful and unsuccessful projects, in addition to differences between author and projects.

III. Blockchain and Software Engineering Methodologies

After being first presented in relation to cryptocurrencies like Bitcoin, blockchain technology has developed into a flexible tool with uses outside of the realm of virtual money. Fundamentally, blockchain is a technology for distributed ledgers that makes it possible for several network members to record and verify transactions in a safe, transparent, and impenetrable manner [16]. Because blockchain technology is decentralised, it does not need middlemen, which promotes trust and facilitates effective peer-to-peer communication [11].

However, there are particular difficulties in designing, creating, and implementing blockchain-based systems, which call for the use of good software engineering practices. Developers of Blockchain have to deal with issues of privacy, security, interoperability, scalability, and performance [9]. Due to blockchain's immutability, maintaining data integrity, consensus processes, and thwarting malevolent assaults become more difficult.

Therefore, in order to overcome these obstacles and provide strong and dependable blockchain solutions, it is essential to use strict SE procedures. Furthermore, blockchain technology is developing quickly, with new frameworks, platforms, and protocols appearing on a regular basis. Because of this dynamic nature, developers must continue to be agile and flexible in their SE methods [16]. As blockchain technology develops and spreads throughout different industries, knowing how developers view SE techniques like collecting requirements, design, testing, and ongoing support can help with the adoption and application of best practices, the detection of possible problems, and the general calibre of software solutions based on blockchain.

3.1 Evaluation of Software Engineering Techniques

Blockchain-based SE has been the subject of research activities, with an emphasis on tackling important issues to improve the creation and implementation of blockchain applications. Agile approaches, which emphasise quick prototyping, iterative development, and strong stakeholder cooperation, have been studied in relation to blockchain projects [17, 19]. The significance of safe coding methods, formal verification, & testing approaches to guarantee the integrity and dependability of smart contracts—

self-executing digital agreements recorded on the blockchain—has also drawn attention to their design and implementation. The steps of requirements collection, development, execution, verification, and maintenance make up a typical SDLC. Conventional approaches to software development, like the waterfall development model, use a step-by-step, linear approach with discrete stages. The waterfall model's rigidity, however, may not be appropriate for the dynamic nature of blockchain technology [14, 16]. In contrast, Agile approaches that emphasise flexibility, teamwork, and incremental development—such as Scrum, Kanban, and extreme software development (XP)—are iterative in nature.

A blockchain is a distributed data structure that can only be appended. It is controlled by a group of interconnected nodes, each of which has a copy of the blockchain and is capable of running SCs, or programs that are stored inside the blockchain. Sending transaction to the network modifies the state of the blockchain; with public blockchains, anybody may submit a transaction [17, 18], but only legitimate ones are handled. The term "blockchain" refers to the sequentially ordered blocks in which the legitimate transactions are recorded. The nodes use a consensus mechanism to create these blocks. One address is used for all transactions, and this address is linked to a private key. Using asymmetric cryptography, transactions originating from an address can only be signed by the private key owner.

When a transaction is reviewed, it may perform one of its public purposes, such as transferring digital money between addresses or creating a SC, in which case all nodes carry out the function. The administration of digital currency or tokens, that possess actual monetary worth, is the focus of the majority of current real-world DApp and smart contract applications [19]. DApps have also been used for various purposes, such as keeping track of identities, voting, games & betting, notarisation of documents, verification of the origin of commodities, and many more. [19, 20].

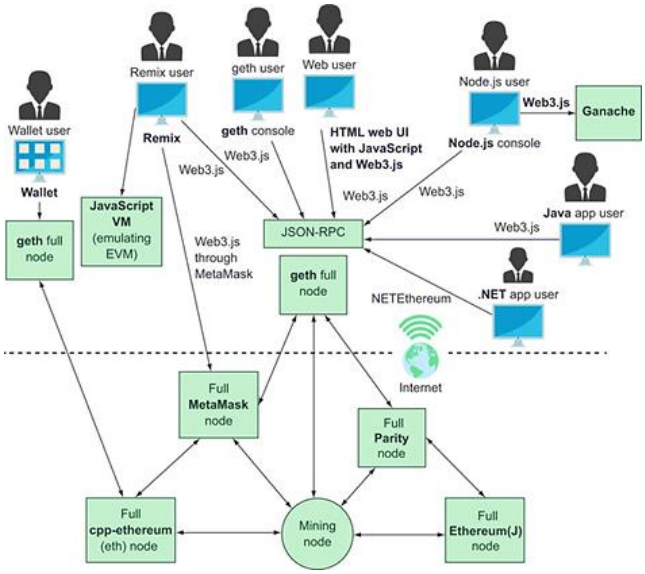

Fig. 1 A typical Ethereum DApp application design.

The App System is illustrated on the right, while blockchain smart contracts are depicted on the left. [20]

3.2 Agility and DApp development

These days, DApp projects throughout the globe have some things in common. Usually, a number of organisations are engaged in initial coin offering (ICO) initiatives that integrate blockchain technologies and raise funds using tokens [22]. Startups are pushing other initiatives in an attempt to capitalise on the novelty of DApps to create game-changing solutions or carve out a niche for themselves. Small, self-organising, co-located teams with highly accessible system requirements specialists are usual in both situations.

DApps are generally not life-critical programs, while some of them may be mission-critical [12]. This is another feature of DApp development. Nonetheless, time-to-market and the capacity to get early user and stakeholder input are crucial since, sometimes, the DApp's initial needs are very loosely specified and prone to change.

DApp development is a perfect fit for Agile Methods (AMs) because of all these characteristics. Actually, small, self-organising teams that may be co-located and working on projects with variable needs are a good fit for AMs [16]. AMs are thought to be capable to produce often and swiftly, which is what DApp applications need [16].

Since Ethereum is currently the most popular blockchain for creating SCs, we will mostly utilise it in the remainder of this work [15]. The Ethereum Virtual Machine (EVM) offers Ethereum nodes that can run appropriate bytecode. Ethereum SCs are really written in Solidity, a specialised high-level language [19]. The typical SC architecture is seen in Fig. 2.

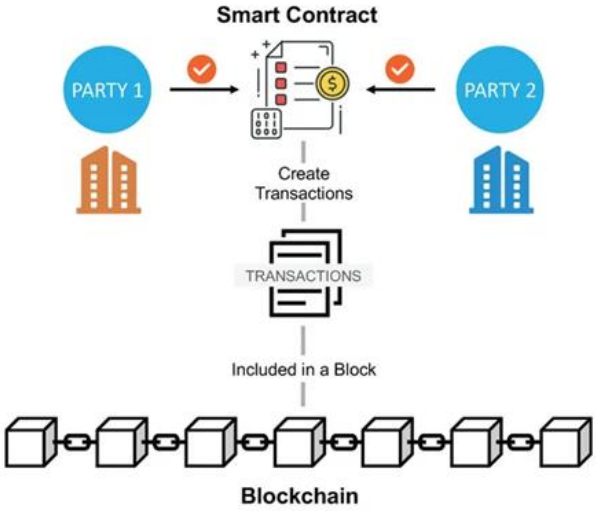

Fig. 2 use the Ethereum Blockchain to run a SC. Every node executes identical bytecode. [16]

IV. Method

The suggested design process for BOS is carried out in a series of phases that are summed up in a UML activity flow diagram (Fig. 3) [19]. To be more specific, the suggested BOS development procedure is as follows:

1. Clearly state the system's objective, summarise it in one or two phrases, & put it wherever all developers can see it. [19, 20].
2. Determine the actors that engage with the system, including external systems and devices as well as human roles.

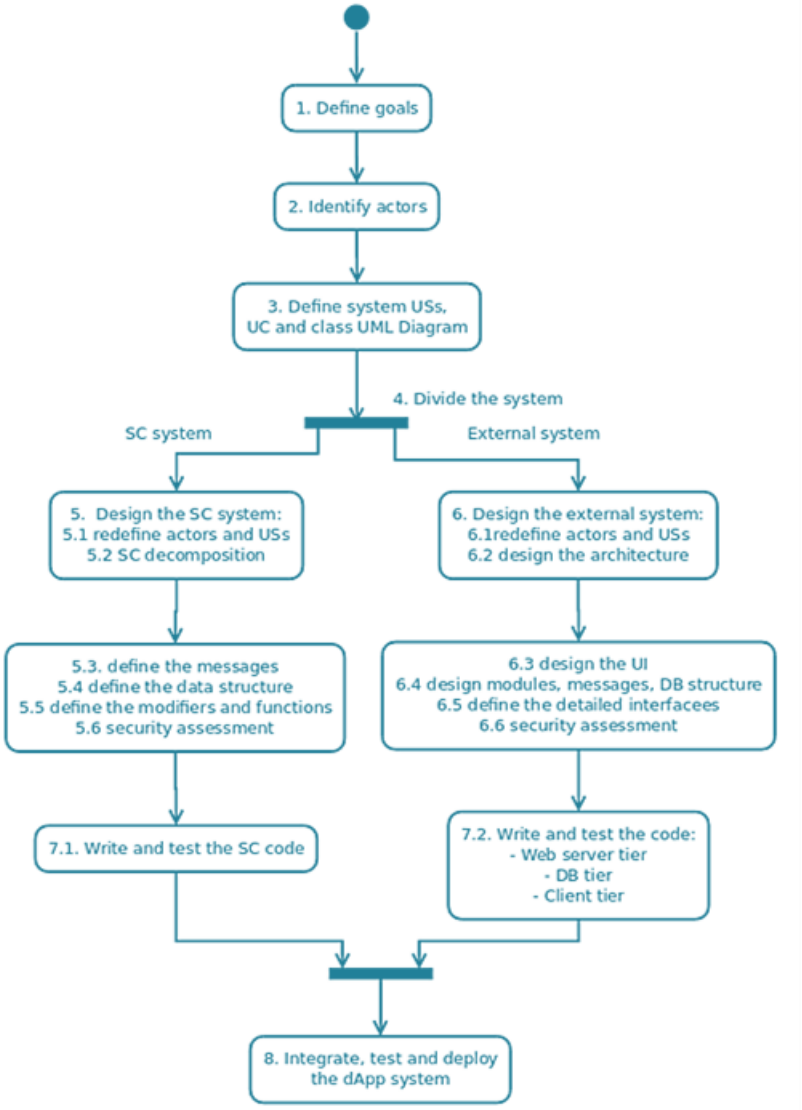
1. Define goals
2. Identify actors
3. Define system USs,
UC and class UML Diagram
4. Divide the system
SC system
External system
5. Design the SC system:
5.1 redefine actors and USs
5.2 SC decomposition
6. Design the external system:
6.1redefine actors and USs
6.2 design the architecture
5.3. define the messages
5.4 define the data structure
5.5 define the modifiers and functions
5.6 security assessment
6.3 design the UI
6.4 design modules, messages, DB structure
6.5 define the detailed interfacees
6.6 security assessment
7.1. Write and test the SC code
7.2. Write and test the code:
- Web server tier
- DB tier
- Client tier
8. Integrate, test and deploy
the dApp system

Fig. 3 The phases in the suggested development process for BOS. [22]

The stereotypes we created to enable the representation of SC ideas in class diagrams created using UML are shown in Table 1 [23]. In addition to the compartments with the name, the attribute, or the functions (operations), there may be another one that represents the events.

Table 1 The addition of stereotypes to the UML class diagram. 26]

Stereotype	Position	Description
«contract»	*Class symbol – upper compartment.*	Denotes a SC.
«Library contract»	*same as above*	A contract taken from some (standard) library.
«struct»	*same as above*	A struct, holding data but no operation, defined and used in the data structure of a contract.
«Enum»	*same as above*	An Enum, holding just a list of possible values «interface» same as above A contract holding only function declarations.
«modifier»	*Class symbol – lower compartment*	A particular kind of function, defined in Solidity.
«array»	*Role of an association*	The 1: n relationship is implemented using an array.
«map»	*same as above*	The 1: n relationship is implemented using a mapping.
«Map[unit]»	*same as above*	The 1: n relationship is implemented using a mapping from integer to the value.

The final three archetypes describe how 1: n connections are implemented in a SC's data structure. The array & the mapping are the two collections that Solidity supports for managing storage, or the data that is permanently kept on the Blockchain. [26].

V. An Example Of Application

Both our university group and the companies we advise with employ the SC development method that has been detailed [26, 27]. A supply-chain administration system, a system to oversee temporary employment contracts, several remote voter registration systems for local government, and a business are a few of the projects under development. Meetings of the board of directors and shareholders.

- **Step 1. Goal of the system:** To oversee proxy delegation and verify the legal number in order to conduct remote voting in corporation assemblies [29].
- **Step 2. Actors:** There are essentially two actors in the system: System management, shareholder and share management, assembly calling, and voting are all handled by the corporate administrator. A person who holds shares may vote, attend assemblies, and designate another shareholder to attend the meetings on his behalf.
- **Step 3. User Stories:** Fig. Using a UML Use Case diagram, and with use cases that are really USs [29], 4 displays the actors and the USs individuals are participating in. Keep in mind that these USs just outline the voting procedure; they are not reliant on the particular technology used to carry it out. They would likewise be correct if a blockchain wasn't used in the implementation.
- As shown, Fig. 4 uses a modified UML diagram of classes to display the SC's data structure. A stakeholder's UML state chart [29, 30] pertaining to their attendance at an assembly is shown in Fig. 4. This UML diagram is utilised exactly as is. It stands for the criteria that proxies be provided before to the commencement of an assembly, that a stakeholder cannot assign another to an assembly after registering, and that a stakeholder may only be granted a certain number of proxies. In this simplified form, it should be noted that delegates cannot be retracted by the delegating shareholder or rejected by the delegated shareholder.

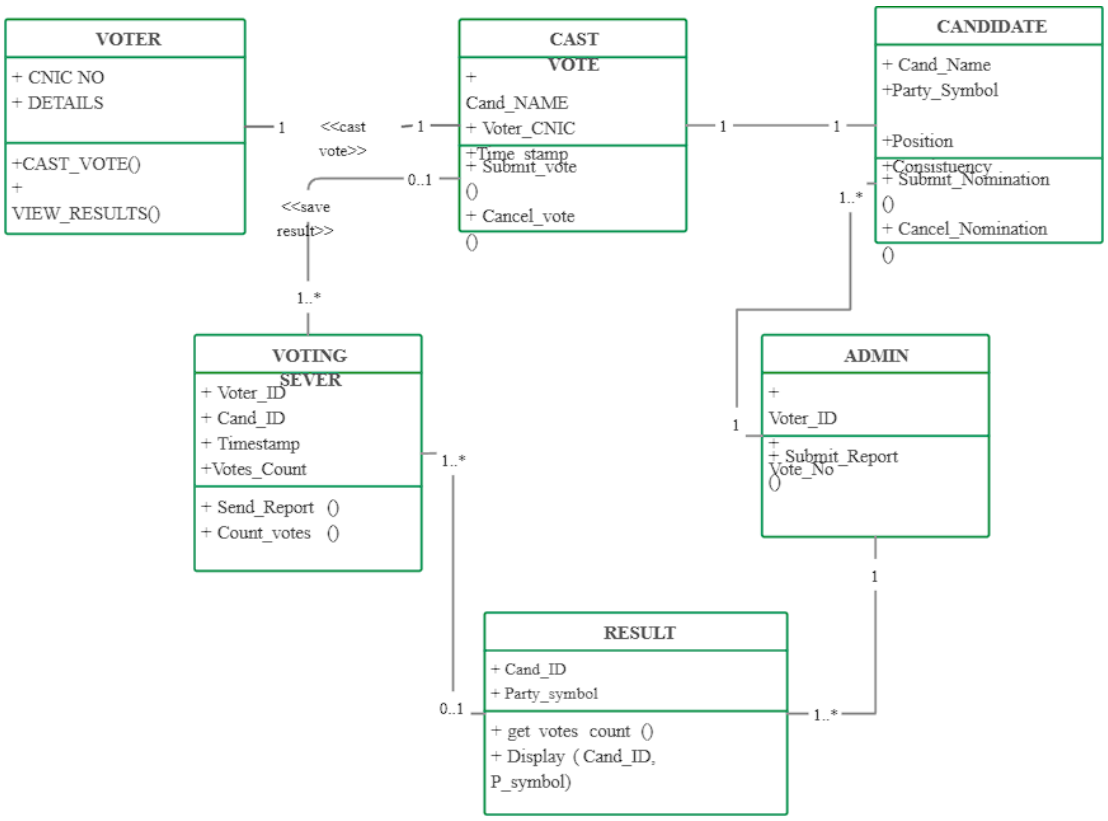

Fig. 4 The altered UML diagram, which displays the needed SC's structure for the voting system. [22, 29]

In this case, the potential UML sequence diagrams that depict the interactions between Actors or SCs are not representational. Actually, every message call takes place between the SC and an Actor (Administrators or Stakeholder) [5]. There are just two people directly interacting in this simple scenario. We don't publish any kind of sequence diagrams because of this [29, 30].

VI. Conclusion

Although a lot of work is now being done to produce D-Apps, software engineering techniques are still not being used well in BOS software development. In actuality, the area is still in its early stages, and academics are still interested in tools or approaches for controlling and modelling the quirks that the

developer of software must deal with while working with blockchain-oriented software systems. Traditional software engineering tools and methods have not yet been adjusted to fit this new software paradigm. Many problems with Blockchain development could be resolved with a good software engineering approach, which would give developers tools like those used in traditional computer science and technology to address security concerns, architectural design, testing plans, and strategies, as well as to enhance software quality or maintenance. Software engineering researchers have a great chance to begin researching a new and very significant area by using established software engineering principles, methods, and instrumentation and adapting and modifying them to this new software technologies. This work goes in this direction by offering a comprehensive modelling of the interactions between the Blockchain environment and traditional software, including class diagrams, state charts, US diagrams, sequence diagrams, and smart contract diagrams, all for BOSE. It also includes a general plan for managing BOS development processes and a working example of a paradigmatic Blockchain smart contract that implements a voting system. Blockchain companies, especially initial coin offerings (ICOs), may benefit greatly from our work by gaining a competitive edge via the use of SE (BOSE) principles from the outset.

References

[1] Faruk, M. J. H., Subramanian, S., Shahriar, H., Valero, M., Li, X., & Tasnim, M. (2022, May). Software engineering process and methodology in blockchain-oriented software development: A systematic study. In *2022 IEEE/ACIS 20th International Conference on Software Engineering Research, Management and Applications (SERA)* (pp. 120-127). IEEE.

[2] Haoyu, W., & Haili, Z. (2012, August). Basic design principles in software engineering. In *2012 Fourth International Conference on Computational and Information Sciences* (pp. 1251-1254). IEEE.

[3] Jones, M., Johnson, M., Shervey, M., Dudley, J. T., & Zimmerman, N. (2019). Privacy-preserving methods for feature engineering using blockchain: review, evaluation, and proof of concept. *Journal of medical Internet research*, *21*(8), e13600.

[4] Mahalakshmi, M., & Sundararajan, M. (2013). Traditional SDLC vs scrum methodology–a comparative study. *International Journal of Emerging Technology and Advanced Engineering*, *3*(6), 192-196.

[5] Marchesi, M., Marchesi, L., & Tonelli, R. (2018, October). An agile software engineering method to design blockchain applications. In *Proceedings of the 14th Central and Eastern European Software Engineering Conference Russia* (pp. 1-8).

[6] Martínez, J. M. G., Carracedo, P., Comas, D. G., & Siemens, C. H. (2022). An analysis of the blockchain and COVID-19 research landscape using a bibliometric study. *Sustainable Technology and Entrepreneurship*, *1*(1), 100006.

[7] Petersen, K., Wohlin, C., & Baca, D. (2009). The waterfall model in large-scale development. In *Product-Focused Software Process Improvement: 10th International Conference, PROFES 2009, Oulu, Finland, June 15-17, 2009. Proceedings 10* (pp. 386-400). Springer Berlin Heidelberg.

[8] Risius, M., & Spohrer, K. (2017). A blockchain research framework: What we (don't) know, where we go from here, and how we will get there. *Business & information systems engineering*, *59*, 385-409.

[9] Porru, S., Pinna, A., Marchesi, M., & Tonelli, R. (2017, May). Blockchain-oriented software engineering: challenges and new directions. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)* (pp. 169-171). IEEE.

[10] Portugal, I., Alencar, P., & Cowan, D. (2018). The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, *97*, 205-227.

[11] Pourheidari, V., Rouhani, S., & Deters, R. (2018, July). A case study of execution of untrusted business process on permissioned blockchain. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (pp. 1588-1594). IEEE.

[12] Singh, S. K., & Kumar, S. (2022). Blockchain Technology: Introduction, Integration, and Security Issues with IoT. In *Applications of Blockchain and Big Iot Systems* (pp. 3-26). Apple Academic Press.

[13] Sultan, K., Ruhi, U., & Lakhani, R. (2018). Conceptualizing blockchains: characteristics & applications. *arXiv preprint arXiv:1806.03693*.

[14] Treude, C. (2022, October). Taming multi-output recommenders for software engineering. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering* (pp. 1-5).

[15] Spilda, F. U., Powell, A. B., Shklovski, I., & Lehuede, S. (2019). Peril V. Promise: IoT and the Ethical Imaginaries.

[16] Wang, W., Hoang, D. T., Hu, P., Xiong, Z., Niyato, D., Wang, P., ... & Kim, D. I. (2019). A survey on consensus mechanisms and mining strategy management in blockchain networks. *Ieee Access*, *7*, 22328-22370.

[17] Xiao, Y., Zhang, N., Lou, W., & Hou, Y. T. (2020). A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, *22*(2), 1432-1465.

[18] Xu, M., Guo, Y., Liu, C., Hu, Q., Yu, D., Xiong, Z., ... & Cheng, X. (2024). Exploring blockchain technology through a modular lens: A survey. *ACM Computing Surveys*, *56*(9), 1-39.

[19] Yan, S. (2022, August). Analysis on blockchain consensus mechanism based on proof of work and proof of stake. In *2022 International Conference on Data Analytics, Computing and Artificial Intelligence (ICDACAI)* (pp. 464-467). IEEE.

[20] Porru, S., Pinna, A., Marchesi, M., & Tonelli, R. (2017, May). Blockchain-oriented software engineering: challenges and new directions. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)* (pp. 169-171). IEEE.

[21] Manifesto, A. (2001). Manifesto for agile software development.

[22] Xu, X., Weber, I., Staples, M., Zhu, L., Bosch, J., Bass, L., ... & Rimba, P. (2017, April). A taxonomy of blockchain-based systems for architecture design. In *2017 IEEE international conference on software architecture (ICSA)* (pp. 243-252). IEEE.

[23] Wessling, F., Ehmke, C., Hesenius, M., & Gruhn, V. (2018, May). How much blockchain do you need? towards a concept for building hybrid dapp architectures. In *Proceedings of the 1st international workshop on emerging trends in software engineering for blockchain* (pp. 44-47).

[24] Fridgen, G., Lockl, J., Radszuwill, S., Rieger, A., Schweizer, A., & Urbach, N. (2018, August). A Solution in Search of a Problem: A Method for the Development of Blockchain Use Cases. In *AMCIS* (Vol. 1, No. 1, pp. 1-11).

[25] Jacobson, L., & Booch, J. R. G. (1999). The unified modeling language reference manual.

[26] Baumeister, H., Koch, N., & Mandel, L. (1999, October). Towards a UML extension for hypermedia design. In *International Conference on the Unified Modeling Language* (pp. 614-629). Berlin, Heidelberg: Springer Berlin Heidelberg.

[27] Yi, X., Wu, D., Jiang, L., Fang, Y., Zhang, K., & Zhang, W. (2022, November). An empirical study of blockchain system vulnerabilities: Modules, types, and patterns. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (pp. 709-721).

[28] Marchesi, L., Marchesi, M., Destefanis, G., Barabino, G., & Tigano, D. (2020, February). Design patterns for gas optimization in ethereum. In *2020 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE)* (pp. 9-15). IEEE.

[29] Baralla, G., Pinna, A., & Corrias, G. (2019, May). Ensure traceability in European food supply chain by using a blockchain system. In *2019 IEEE/ACM 2nd International Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)* (pp. 40-47). IEEE.

[30] Wang, R. C. A. I. Empirical Methods and Studies in Software Engineering.